EUV IMAGING SPECTROMETER
Hinode

EIS SOFTWARE NOTE No. 22

Version 1.2 2017 Sep 13

EIS Quicklook Data Processing and Timeline Database Retrieval

John Mariska
George Mason University
Department of Physics and Astronomy
4400 University Drive
Fairfax VA 22030
USA

jtmariska@gmail.com

1 Overview

The purpose of this software note is to document the software that manages EIS quicklook data processing. This should allow individuals other than the author to modify the code should that be necessary. It also documents the method used to transfer the EIS planning database files from ISAS to NRL and from there to the EIS SolarSoft distribution.

2 Quicklook data processing

Quicklook data processing takes place on the reformatting computers at ISAS. Since the available data are often incomplete, the software regularly runs into difficulty and either crashes or hangs, making it difficult to automate the task. To overcome this problem, the program eis_do_quicklook.py manages quicklook data processing. This program constructs the necessary command string to spawn to the operating system and then monitors the resulting process. If it detects any of a number of known failure modes, it takes appropriate action. If the program detects an unknown failure mode, it tries restarting the process before finally giving up after 15 attempts. The quicklook data files produced are automatically copied to the appropriate locations on DARTS, and leftover files are regularly purged from the working directories. Note that if the program needs to restart many times, it can take up to an hour to run to completion.

The program runs daily at 23:05 JST via a cron task that executes the shell script eis_quicklook.csh in the $HOME/bin directory on the reformatting computer. At the present time the program must be run in the directory $HOME/work/localdata/sdtp/hpw/decomp_new, and the shell script takes care of that as well as copying the log file that the program produces to a web-accessible location on DARTS (`http://darts.isas.jaxa.jp/pub/solar/solarb/eis/staging/logs/quicklook/`).

Generally, the cron task retrieves the most recent data in a timely manner. Under exceptional circumstances, the program can be run manually, either by invoking the cron script in the $HOME/bin directory or by going to the directory where the program resides and executing the python file. The documentation at the top of the listing in §2.2 shows the format for invoking the program. Note that the date must be entered in the format shown, **dd-mon-yyyy**, where mon may be either the three letter abbreviation for the month on the two digit number for the month.

The quicklook reformatting software can be run by using ssh to connect from ope-gw to rfsb1.reformat.isas.jaxa.jp (133.74.197.31) as user sbukeis. Contact an ISAS system administrator to arrange for access.

2.1 Shell script Listing

```
#!/bin/csh

# (2009-Jun-25) JTM: initial test version
# (2012-Aug-30) JTM: added copy of timeline to darts
# (2012-Dec-11) JTM: moved timeline copy to own script
# (2017-Sep-13) JTM: logfile name change

cd $HOME/work/localdata/sdtp/hpw/decomp_new
set logfile = quicklook_log_‘date ’+%Y%m%d_%H%M’‘.txt
./eis_do_quicklook.py >& $logfile
if (! -z $logfile) then
```

```
    cp $logfile /soda/solarb/eis/staging/logs/quicklook
endif

# Do a little cleanup
cd /soda/solarb/eis/staging/logs/quicklook
find . -name 'quicklook*.txt' -mtime +10 -exec rm -f {} \;
```

2.2 Program Listing

```
#!/usr/bin/env python

"""
eis_do_quicklook.py - automated quicklook processing

    One Program to rule them all, One Program to find them,
    One Program to bring them all and in the darkness bind them
    In the Land of ISAS where the Shadows lie.

SYNOPSIS

    ./eis_do_quicklook.py [dd-mon-yyyy]

DESCRIPTION

    This Python script runs the EIS quicklook processing software for
    either the selected day or by default the previous day. The
    created FITS files are copied to the appropriate area on DARTS. In
    its present form, this script can only be run from this
    directory. The script is normally invoked by the cron script
    ~/bin/eis_quicklook.csh. When that script is used, a log file is
    written to the EIS staging area on DARTS:

    (http://darts.isas.jaxa.jp/pub/solar/solarb/eis/staging\
    /logs/quicklook).

    The script attempts to recover from a number of failure modes
    exhibited by the quicklook software. If for some reason it
    discovers a new failure mode, the script will give up after
    restarting the quicklook software 15 times.

EXAMPLES

    ./eis_do_quicklook.py 12-Jun-2016

    ./eis_do_quicklook.py 12-06-2016

    ./eis_do_quicklook.py

TODO

    Add the ability to run from any directoy.

HISTORY

    (2009-Jun-24) JTM: Initial production version.
    (2010-Mar-03) JTM: Additional cleanup.
    (2014-Sep-02) JTM: Added kill_subs call when restarts too many times.
    (2017-Jan-11) JTM: Make date handling more robust.

CONTACT

    John Mariska, jmariska@gmu.edu, jtmariska@gmail.com
"""

import os
import sys
```

```
import time
from dateutil import parser
import commands
import shutil


def mk_batch_script():
    cmd1 = "eis_quick_look, '%s', /get_data\n\n" % (day)
    cmd2 = "eis_quick_look, '%s', /restart\n\n" % (day)

    ip = open('eis_do_quicklook_in.txt', mode='w')
    ip.write('; eis_do_quicklook_in.txt\n')
    ip.write('; count: %d\n' % (count))
    ip.write('; ' + time.asctime() + '\n\n')
    ip.write('print, "* * * Batch start * * *"\n\n')
    if restart:
        ip.write(cmd2)
    else:
        ip.write(cmd1)
    ip.write('print, "* * * Batch end * * *"\n\n')
    ip.write('exit\n')
    ip.close()


def kill_subs(pid):
    # For now, assume there are just two processes below pid to
    # worry about, the SSW script and the IDL process. This could
    # be a mess if there are more.
    pscmd = 'ps --format pid --no-headers --ppid %s'
    kstring = '/usr/bin/kill -s KILL '

    pid1 = commands.getoutput(pscmd % (str(pid)))
    if pid1:
        pid2 = commands.getoutput(pscmd % (pid1))
    else:
        pid2 = ''

    # Kill in reverse order one at a time for now
    print 'Killing processes %s %s %s' % (str(pid), pid1, pid2)
    if pid2: os.system(kstring + pid2)
    if pid1: os.system(kstring + pid1)
    if str(pid): os.system(kstring + str(pid))

    # Remove IDL start script
    sswfile = '/home/sbukeis/ssw_idl.' + pid1.strip()
    if os.path.exists(sswfile):
        print 'Removing ' + sswfile
        os.remove(sswfile)


# * * * M A I N * * *

print 'Started at ', time.asctime()

if len(sys.argv) == 2:
    dt = parser.parse(sys.argv[1], dayfirst=True)
    day = dt.strftime('%d-%b-%Y')
else:
    yesterday = time.time() - 24*60*60 - 4*60*60 # adjust as needed for cron use
    day = time.strftime('%d-%b-%Y', time.localtime(yesterday))

# Remove any leftover push scripts
if os.path.exists('push_files.sh'): os.remove('push_files.sh')

# First script gets data
count = 1
maxcount = 15
restart = False
```

```
mk_batch_script()

pid = os.spawnlp(os.P_NOWAIT, './eis_do_quicklook.sh', 'eis_do_quicklook.sh')
print 'PID: ', pid

tlim = 90.0
len = 0
while True:
    time.sleep(tlim)

    newlen = os.stat('eis_do_quicklook_out.txt').st_size
    print time.asctime()
    print 'Size: ', newlen
    line = commands.getoutput('tail -1 eis_do_quicklook_out.txt')

    if newlen > len: # still growing, keep going
        len = newlen
        continue
    elif count > maxcount: # may be stuck in a loop
        print 'Too many restarts. Aborting.'
        kill_subs(pid)
        break
    elif line == '* * * Batch end * * *':
        if os.path.exists('push_files.sh'): # really are done!
            print 'Completed'
            print 'Script called %d times' % (count)
            pushlen = os.stat('push_files.sh').st_size
            if pushlen > 0:
                print 'Running push_files.sh'
                os.system('./push_files.sh')
            else:
                print 'Problem with push_files.sh: zero length'
            break
        else: # might have crashed back to shell, restart
            len = 0 # outfile started again
            restart = True
            count = count + 1
            mk_batch_script()
            pid = os.spawnlp(os.P_NOWAIT, './eis_do_quicklook.sh',
                             'eis_do_quicklook.sh')
            print 'Restarting after apparent IDL exit'
            print 'New PID: ', pid
            continue
    elif line == 'Segmentation fault': # crashed back to shell, restart
        len = 0 # outfile started again
        restart = True
        count = count + 1
        mk_batch_script()
        pid = os.spawnlp(os.P_NOWAIT, './eis_do_quicklook.sh', 'eis_do_quicklook.sh')
        print 'Restarting after IDL exit on Segmentation fault'
        print 'New PID: ', pid
        continue
    else: # must be stalled, restart
        kill_subs(pid)
        len = 0 # outfile started again
        restart = True
        count = count + 1
        mk_batch_script()
        pid = os.spawnlp(os.P_NOWAIT, './eis_do_quicklook.sh', 'eis_do_quicklook.sh')
        print 'Restarting after stalling'
        print 'New PID: ', pid
        continue

# Clean up old directories and log files
ndays = 10
day_to_rm = time.time() - ndays*24*60*60
day_to_rm_str = time.strftime('eis_fits_%Y%m%d', time.localtime(day_to_rm))
if os.path.exists(day_to_rm_str): shutil.rmtree(day_to_rm_str)
```

```
lfile = time.strftime('quicklook_log_%Y%m%d.txt', time.localtime(day_to_rm))
if os.path.exists(lfile): os.remove(lfile)

dir_str = time.strftime('%Y%m%d', time.localtime(day_to_rm))
full_path = '/home/sbukeis/work/eis/localdata/sdtp/md/' + dir_str
if os.path.exists(full_path): shutil.rmtree(full_path)

full_path = '/home/sbukeis/work/eis/localdata/sdtp/fits/mission/' + dir_str
if os.path.exists(full_path): shutil.rmtree(full_path)

print 'Finished at ', time.asctime()
print 'Restarted %d times' % (count - 1)
```

3 Timeline database retrieval

The EIS timeline data in Japan are handled in a somewhat peculiar manner. The master timeline resides on the EIS planning computer at ISAS. It is either generated there or copied to that computer by the remote planner. Because of security concerns, the files can not be rsynced from that computer to NRL or Oslo. The planning computer, however, can be reached from the reformatting computers and those computers can write files to a publicly available directory on DARTS. The final task run by the planner on the ISAS EIS planning computer (∼/scripts/db_backup.sh) includes a copy of the planning database files to a directory on the planning computer that is away from the standard storage location (∼/mariska/timeline). Once a day at 19:35 JST, the cron task listed below on the reformatting computer then copies the files from there to DARTS. The timeline files are located at `http://darts.isas.jaxa.jp/pub/solar/solarb/eis/staging/timeline`. This is **very** kludgey, but has worked well for quite some time.

```
#!/bin/csh

# eis_timeline.csh
# (2012-Dec-11) JTM: copy timeline to darts

# copy timeline to darts
# Note master site is $HOME/eisco/planning_db/timeline_db.
# This may be safer, since no poss of user doing planning on it.
cp -rp $HOME/eisco/mariska/timeline/* /soda/solarb/eis/staging/timeline
```

A Revision History

Revision	Date	Author(s)	Description
1.0	2016 May 18	John Mariska	Creation
1.1	2017 Jan 12	John Mariska	More robust date handling in code and additional info on running the software.
1.2	2017 Sep 13	John Mariska	Minor modification to shell script so that it can be run more than once a day.